



CUDA PROFILING TOOLS OVERVIEW

JACKSON MARUSARZ

NSIGHT DEVELOPER TOOLS

CUDA Tools

OptiX Tools

Graphics Tools

CUDA Debugging
CUDA Build Mgt
CUDA Code Editing

Nsight Visual
Studio Edition

Nsight Visual
Studio Code
Edition

Nsight Eclipse
Edition

cuda-gdb

Compute
Sanitizer

Correctness Profiler
Memory Checker

CUDA Kernel Profiling
CUDA API Debugging
OptiX API Debugging

Nsight Compute

System Trace
CUDA API+GPU
CUDA-X & Magnum IO
OpenGL, DirectX, Vulkan
RTX Raytracing, OptiX API
Multi-GPU
GPU Metrics sampling
Frame Stutter Analysis

Nsight Systems

Nsight Deep
Learning Designer

Inferencing Code Gen
Feature Map Debug
DNN Profiling
DNN Model Editor

Nsight Graphics

API Debugging
Graphics Profiling
RTX Raytracing
Shader Profiling
C++ Frame Capture

Profile & Trace API

CUPTI

Annotation API

NVTX

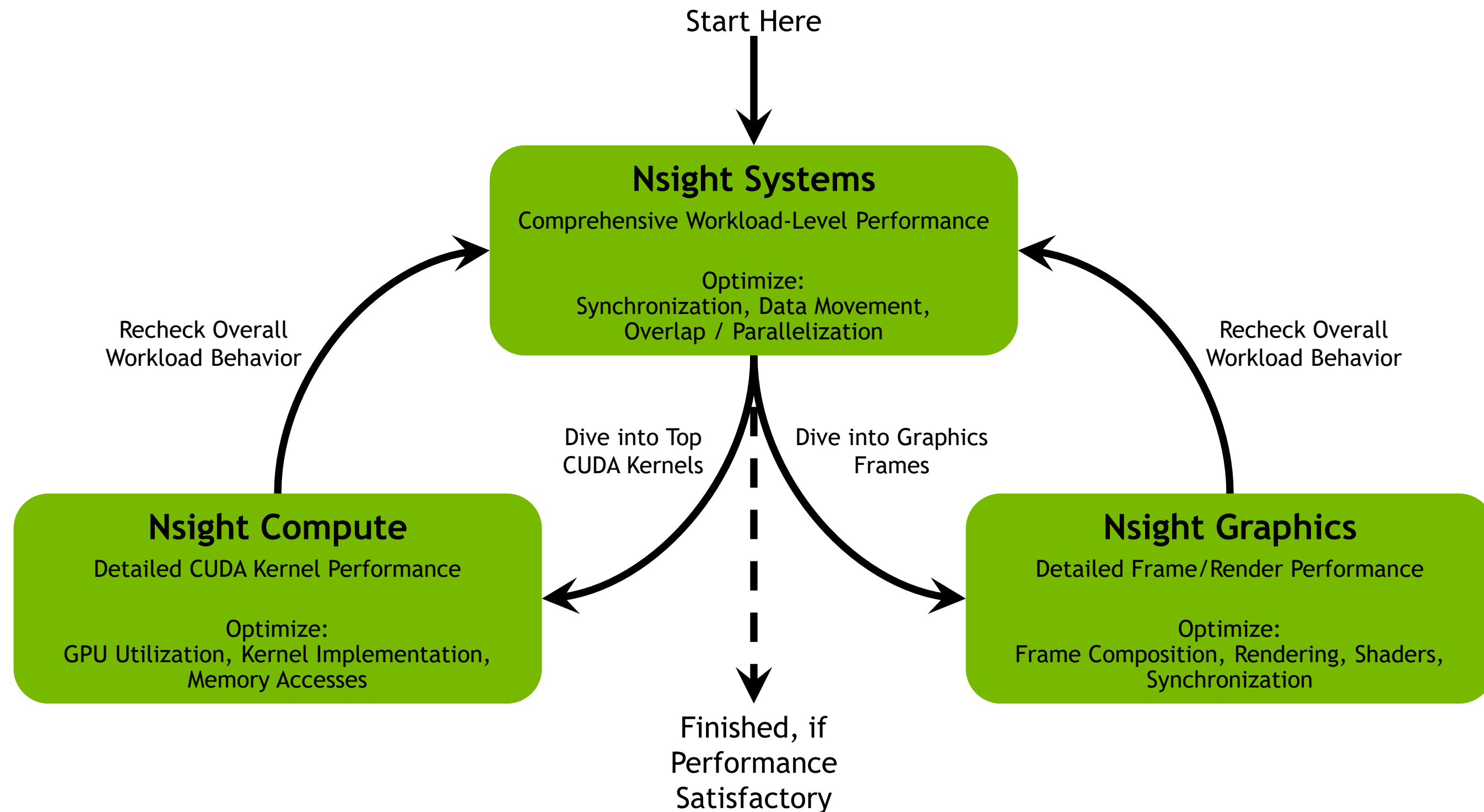
Nsight Aftermath
SDK

GPU Crash Dump

Nsight Perf SDK

GPU Perf Metric Recorder
HTML Report Generation

NSIGHT PROFILING TOOLS WORKFLOW





NSIGHT SYSTEMS



NSIGHT SYSTEMS

System Profiler

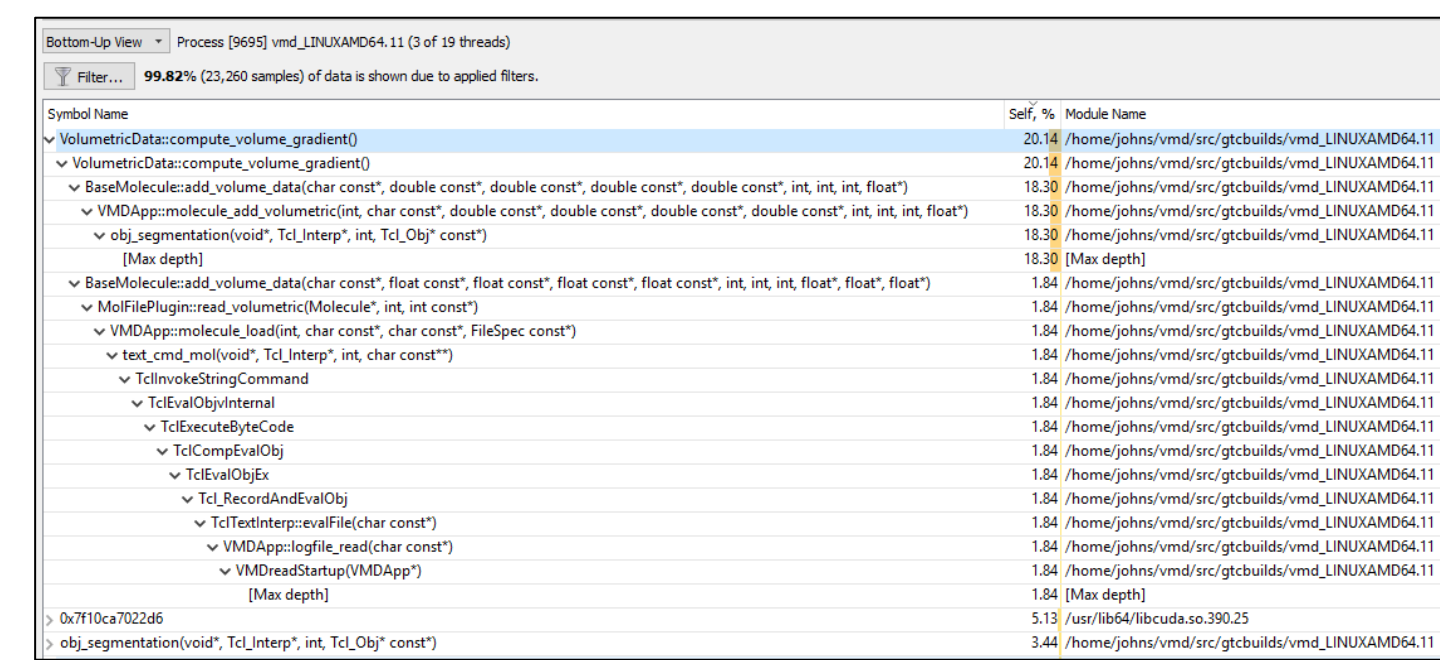
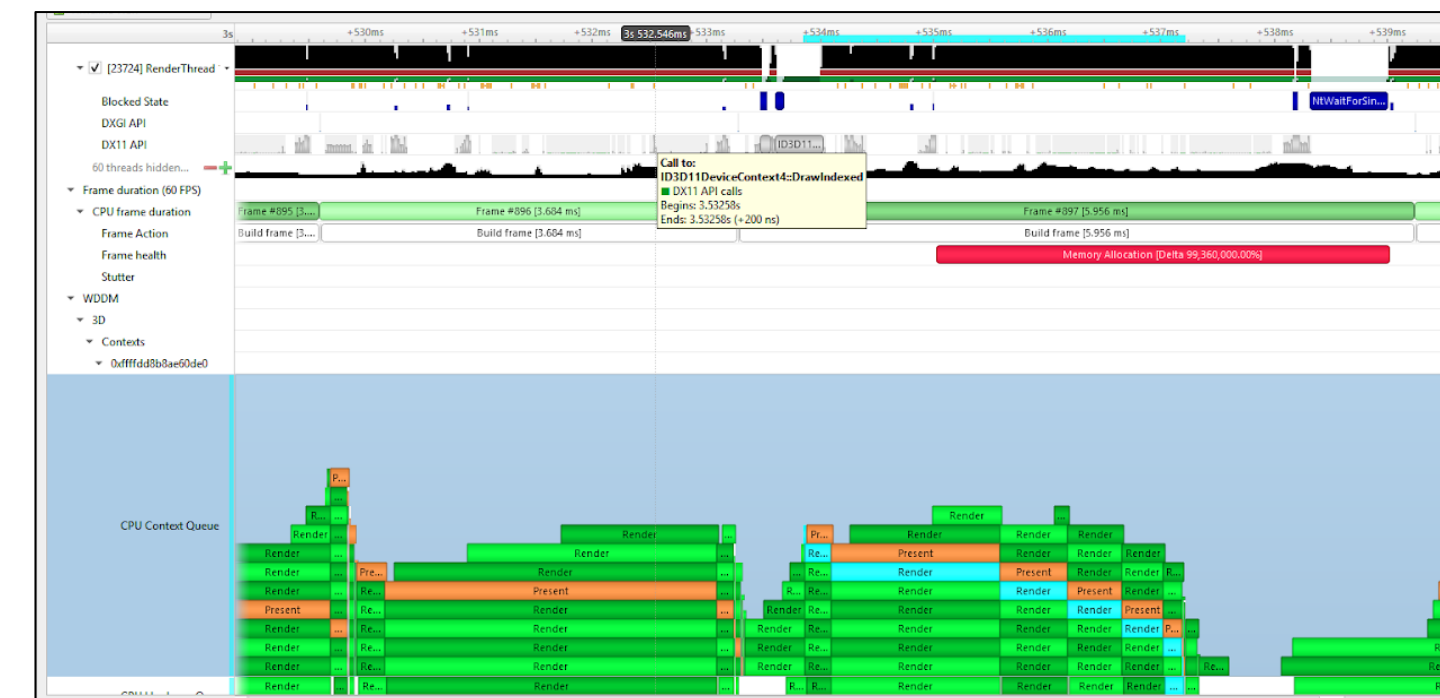
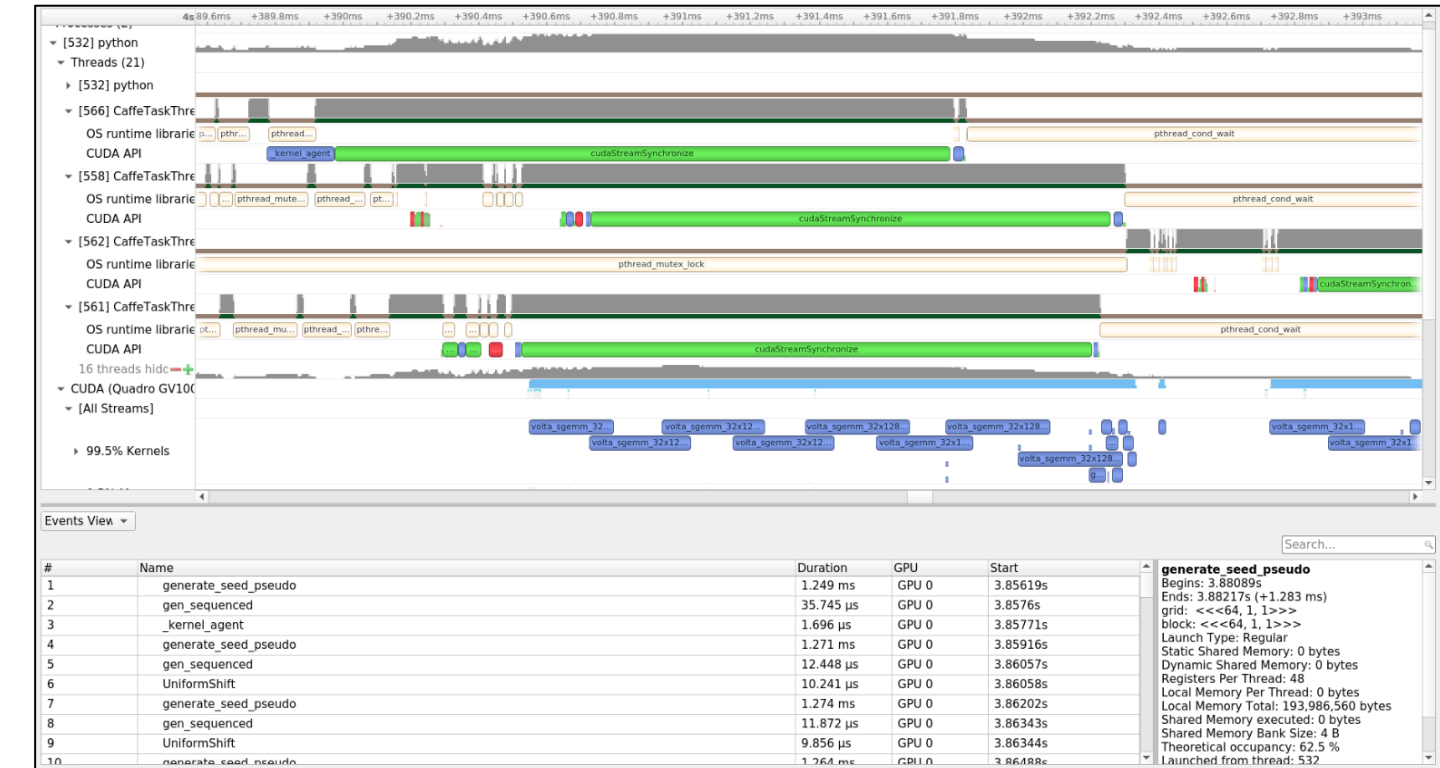
Key Features:

- System-wide application algorithm tuning
 - Multi-process tree support
- Locate optimization opportunities
 - Visualize millions of events on a very fast GUI timeline
 - Or gaps of unused CPU and GPU time
- Balance your workload across multiple CPUs and GPUs
 - CPU algorithms, utilization and thread state
 - GPU streams, kernels, memory transfers, etc
- Command Line, Standalone, IDE Integration

OS: Linux (x86, Power, Arm SBSA, Tegra), Windows, MacOSX (host)

GPUs: Pascal+

Docs/product: <https://developer.nvidia.com/nsight-systems>



Processes & threads

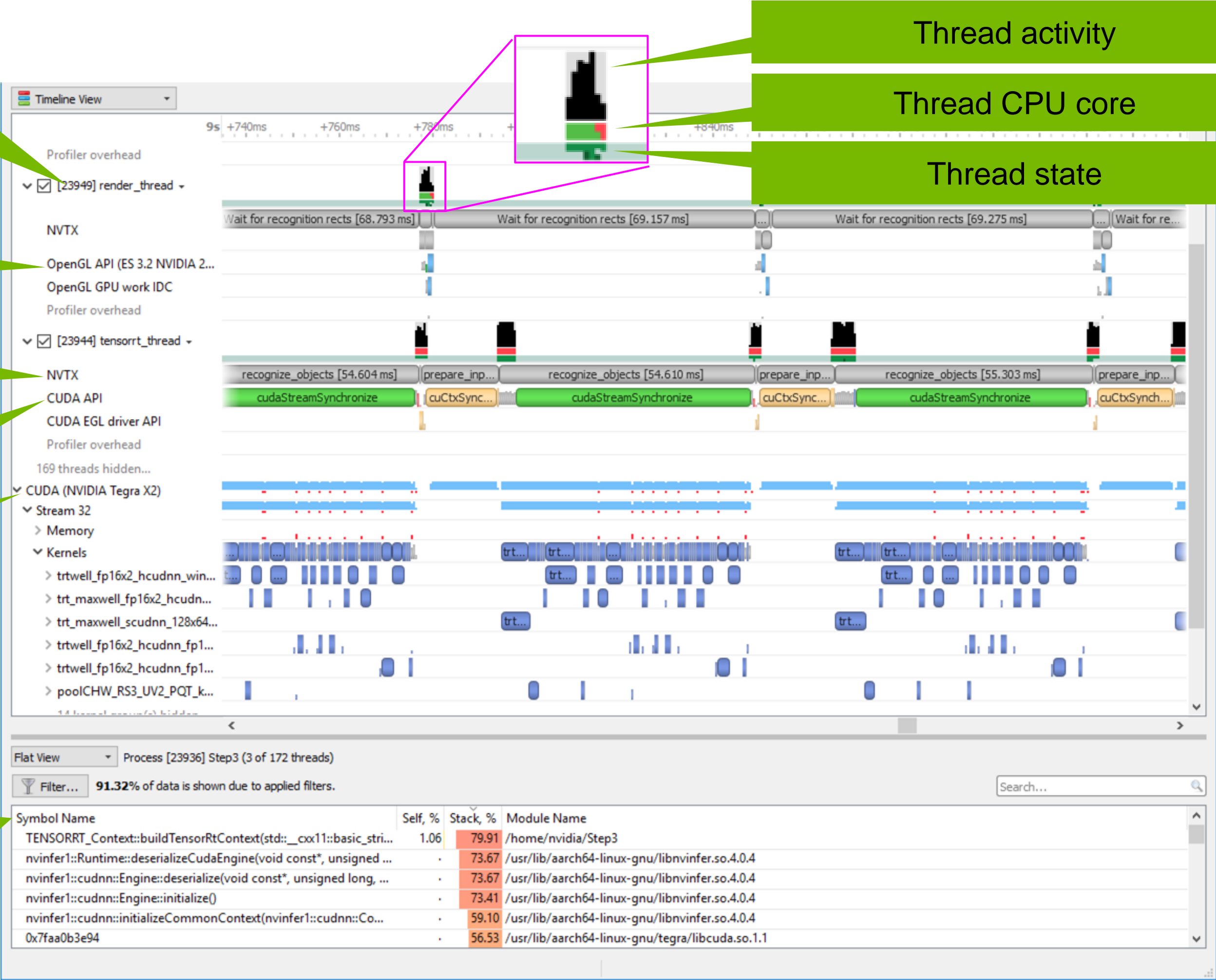
OpenGL trace

NVTX annotations

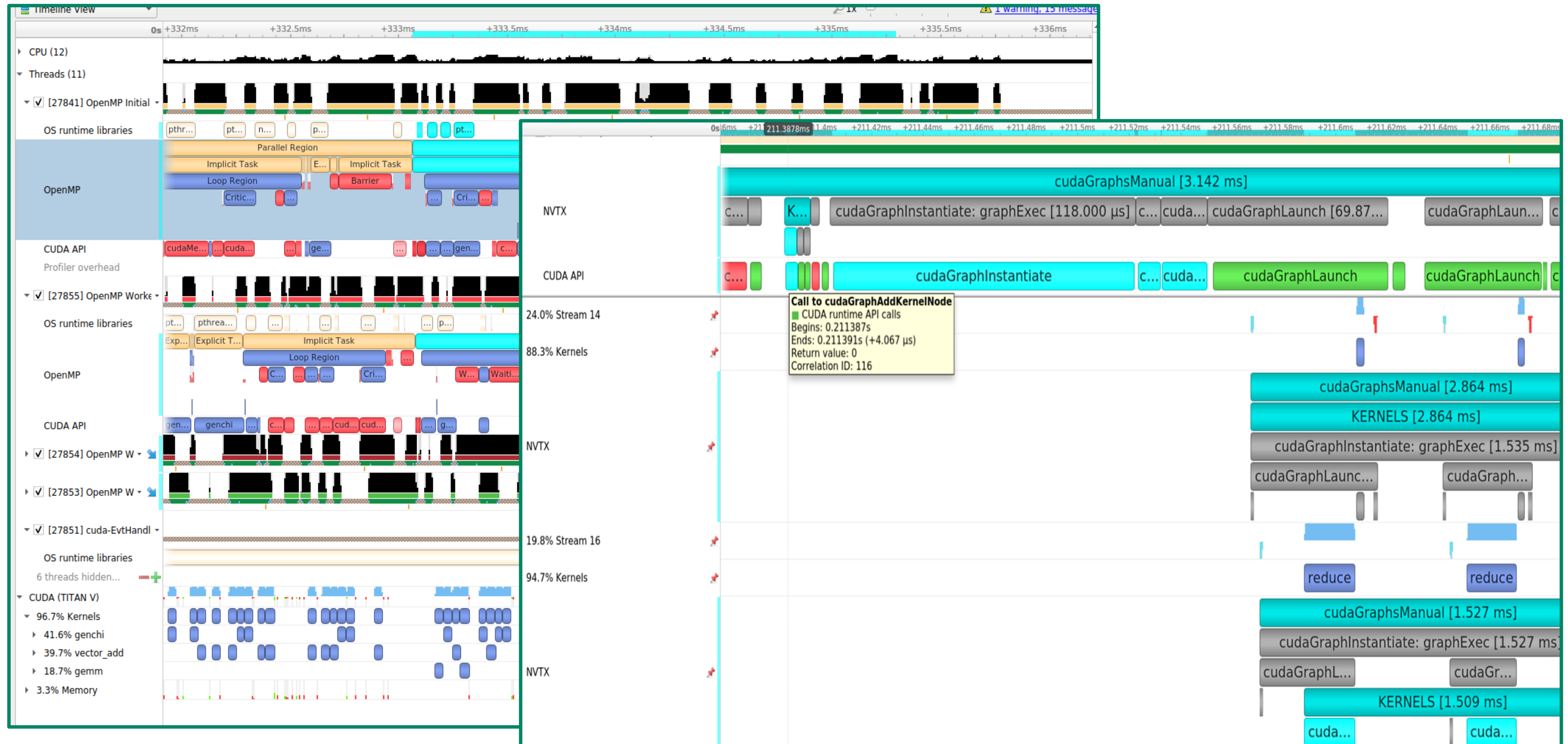
CUDA API trace

GPU CUDA
Kernel & memory
transfer activities

CPU call-stack samples



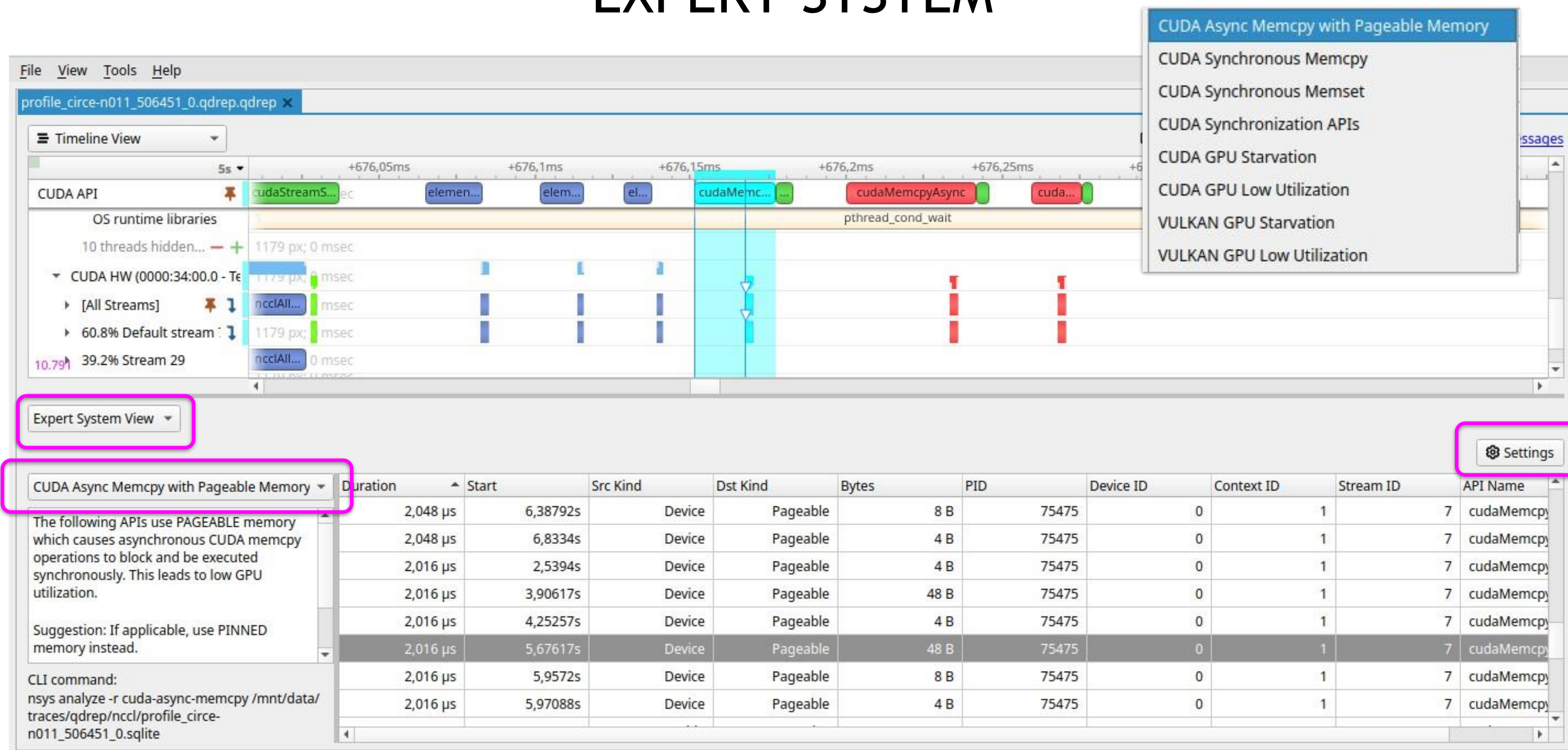
ZOOM/FILTER TO EXACT AREAS OF INTEREST



FUNCTION TABLE WITH CALL-STACK BACKTRACES

Bottom-Up View ▾ Process [9695] vmd_LINUXAMD64.11 (3 of 19 threads)		
Filter... 99.82% (23,260 samples) of data is shown due to applied filters.		
Symbol Name	Self, %	Module Name
▼ VolumetricData::compute_volume_gradient()	20.14	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
▼ VolumetricData::compute_volume_gradient()	20.14	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
▼ BaseMolecule::add_volume_data(char const*, double const*, double const*, double const*, double const*, int, int, int, float*)	18.30	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
▼ VMDApp::molecule_add_volumetric(int, char const*, double const*, double const*, double const*, double const*, int, int, int, float*)	18.30	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
▼ obj_segmentation(void*, Tcl_Interp*, int, Tcl_Obj* const*)	18.30	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
[Max depth]	18.30	[Max depth]
▼ BaseMolecule::add_volume_data(char const*, float const*, float const*, float const*, float const*, int, int, int, float*, float*, float*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
▼ MolFilePlugin::read_volumetric(Molecule*, int, int const*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
▼ VMDApp::molecule_load(int, char const*, char const*, FileSpec const*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
▼ text_cmd_mol(void*, Tcl_Interp*, int, char const**)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
▼ TclInvokeStringCommand	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
▼ TclEvalObjvInternal	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
▼ TclExecuteByteCode	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
▼ TclCompEvalObj	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
▼ TclEvalObjEx	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
▼ Tcl_RecordAndEvalObj	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
▼ TclTextInterp::evalFile(char const*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
▼ VMDApp::logfile_read(char const*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
▼ VMDreadStartup(VMDApp*)	1.84	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11
[Max depth]	1.84	[Max depth]
> 0x7f10ca7022d6	5.13	/usr/lib64/libcuda.so.390.25
> obj_segmentation(void*, Tcl_Interp*, int, Tcl_Obj* const*)	3.44	/home/johns/vmd/src/gtcbuilds/vmd_LINUXAMD64.11

EXPERT SYSTEM



GPU METRICS SAMPLING

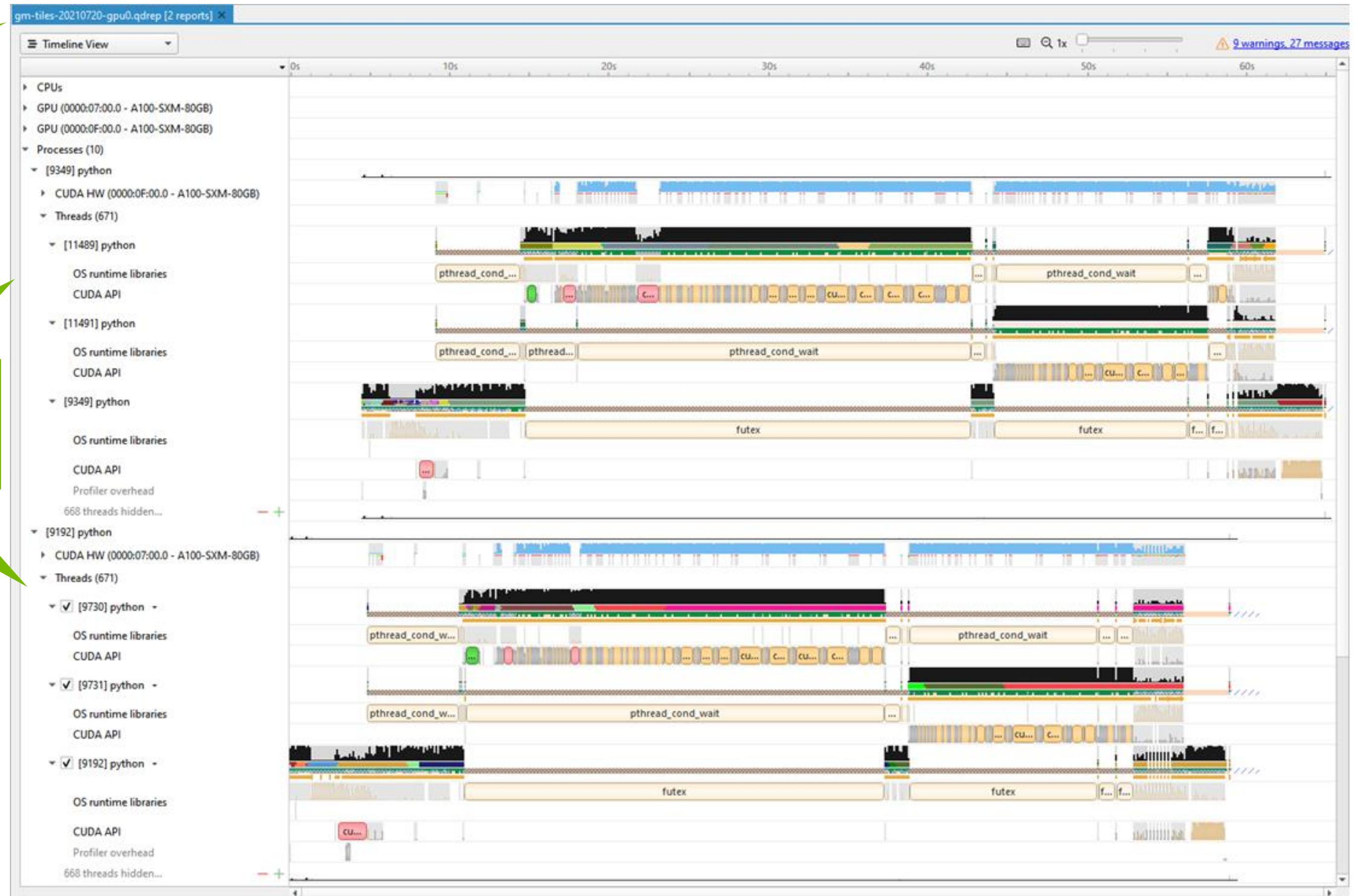


MULTI-REPORT TILING

Visualize more parallel activity

Open multiple reports

Correlated on same timeline



The background is a dark, almost black, field filled with a complex network of thin, glowing green lines. These lines intersect and connect various points, some of which are marked by small, bright green dots. There are also a few larger, faint blue circular shapes scattered across the background, adding to the abstract, digital feel of the image.

NSIGHT COMPUTE



NSIGHT COMPUTE

Kernel Profiling Tool

Key Features:

- Interactive CUDA API debugging and kernel profiling
- Built-in rules expertise
- Fully customizable data collection and display
- Command Line, Standalone, IDE Integration, Remote Targets

OS: Linux (x86, Power, Tegra, Arm SBSA), Windows, MacOSX
(host only)

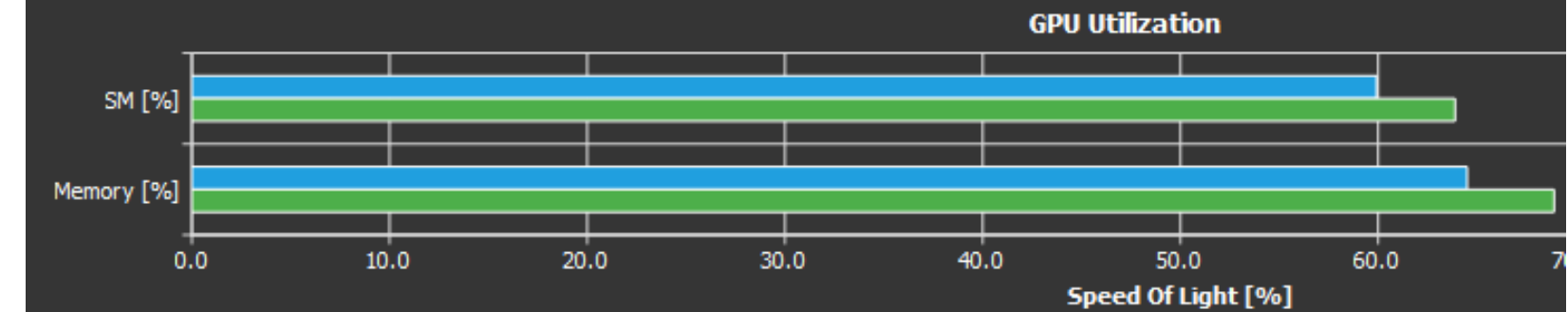
GPUs: Volta+

Docs/product: <https://developer.nvidia.com/nsight-compute>

GPU Speed Of Light

High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of the theoretical maximum. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.

SOL SM [%]	59.93	(-6.20%)	Duration [usecond]
SOL Memory [%]	64.50	(-6.38%)	Elapsed Cycles [cycle]
SOL L1/TEX Cache [%]	26.92	(-5.33%)	SM Active Cycles [cycle]
SOL L2 Cache [%]	64.50	(-6.38%)	SM Frequency [cycle/nsecond]
SOL DRAM [%]	51.55	(+84.34%)	DRAM Frequency [cycle/nsecond]



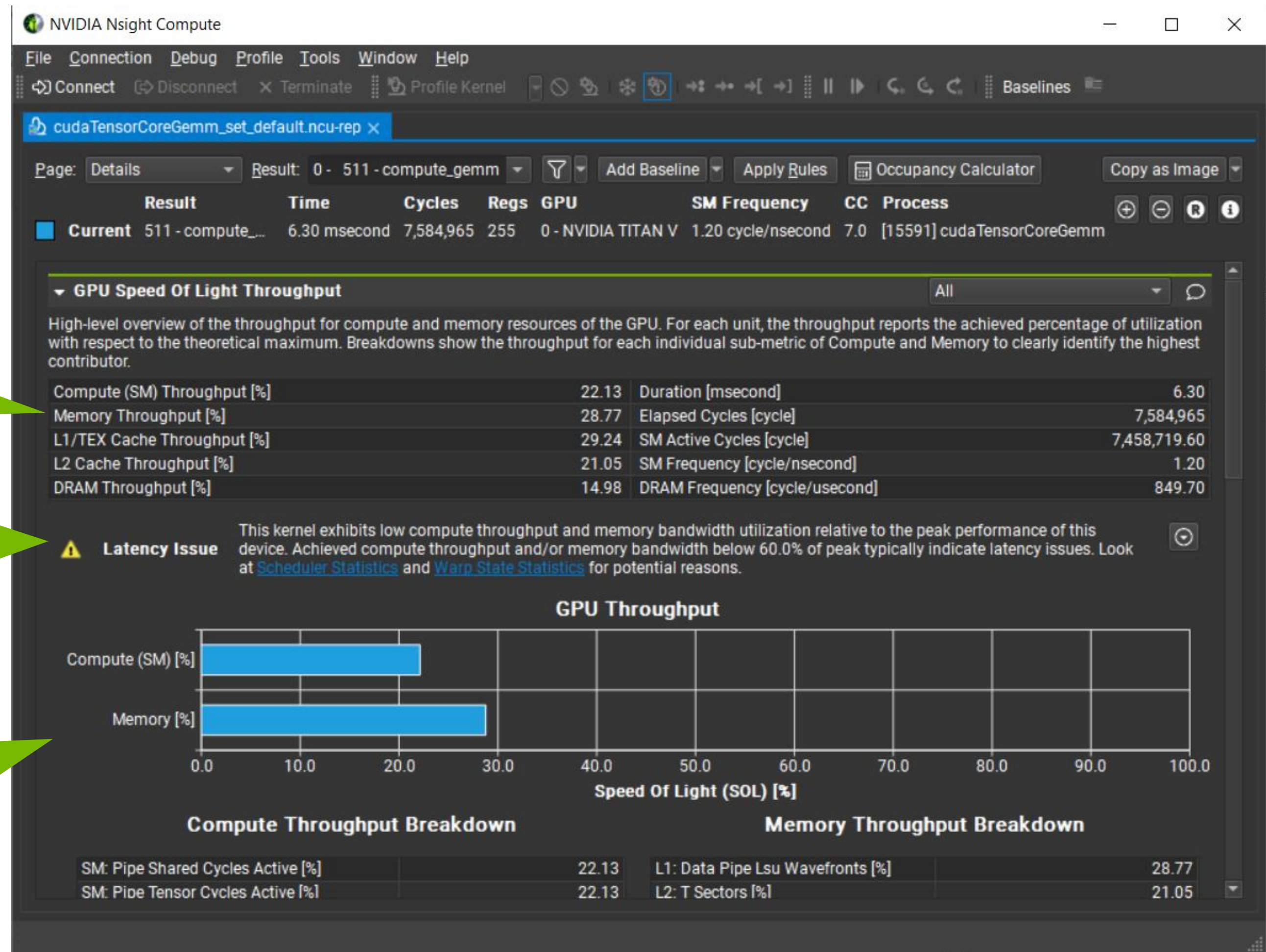
inst_executed [inst]	63,021,056 (284 instances)
l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_ld.sum	0
l1tex__data_bank_conflicts_pipe_lsu_mem_shared_op_st.sum	0
l1tex__data_bank_reads.avg.pct_of_peak_sustained_elapsed [%]	9.66
l1tex__data_bank_writes.avg.pct_of_peak_sustained_elapsed [%]	3.23
l1tex__data_pipe_lsu_wavefronts.avg.pct_of_peak_sustained_elapsed [%]	46.16
l1tex__data_pipe_lsu_wavefronts_mem_shared_cmd_read.sum	25,165,824
l1tex__data_pipe_lsu_wavefronts_mem_shared_cmd_read.sum.pct_of_peak_sustained_active [%]	40.75
l1tex__data_pipe_lsu_wavefronts_mem_shared_cmd_write.sum	2,097,152
l1tex__data_pipe_lsu_wavefronts_mem_shared_cmd_write.sum.pct_of_peak_sustained_active [%]	3.40
l1tex__data_pipe_tex_wavefronts.avg.pct_of_peak_sustained_elapsed [%]	0
l1tex__f_wavefronts.avg.pct_of_peak_sustained_elapsed [%]	0.00
l1tex__lsu_writeback_active.avg.pct_of_peak_sustained_elapsed [%]	42.59
l1tex__lsu_writeback_active.sum [cycle]	27,803,648
l1tex__lsu_writeback_active.sum.pct_of_peak_sustained_active [%]	45.03
l1tex__lsuin_requests.avg.pct_of_peak_sustained_elapsed [%]	66.00
l1tex__m_l1tex2xbar_req_cycles_active.avg.pct_of_peak_sustained_elapsed [%]	3.40
l1tex__m_l1tex2xbar_write_bytes.sum [Mbyte]	4.19
l1tex__m_l1tex2xbar_write_bytes_mem_global_op_red.sum [byte]	0

@P0 EXIT	6	108	49	1,404,672
IADD3 R7, P2, R0, UR7, RZ	7	177	95	1,401,344
IADD3 R6, P1, R4, UR4, RZ	8	30	0	1,401,344
ISETP.GE.U32.AND P0, PT, R7, UR5, PT	8	14	0	1,401,344
IADD3.X R8, R2, UR8, RZ, P2, !PT	8	14	0	1,401,344
IMAD.X R7, RZ, RZ, R5, P1	9	9	0	1,401,344
ISETP.GE.U32.AND.EX P0, PT, R8, UR6, PT, P0	9	106	35	1,401,344
STG.E.U8 [R6.64], R3	8	116	75	1,401,344
@P0 EXIT	8	92	33	1,401,344
IADD3 R8, P2, R0, UR9, RZ	9	45	14	1,397,120
IADD3 R6, P1, R6, UR4, RZ	9	248	145	1,397,120
ISETP.GE.U32.AND P0, PT, R8, UR5, PT	9	57	17	1,397,120
IADD3.X R8, R2, UR12, RZ, P2, !PT	9	14	4	1,397,120
IMAD.X R7, RZ, RZ, R7, P1	9	7	0	1,397,120
ISETP.GE.U32.AND.EX P0, PT, R8, UR6, PT, P0	9	94	15	1,397,120
STG.E.U8 [R6.64], R3	8	104	61	1,397,120

Targeted metric sections

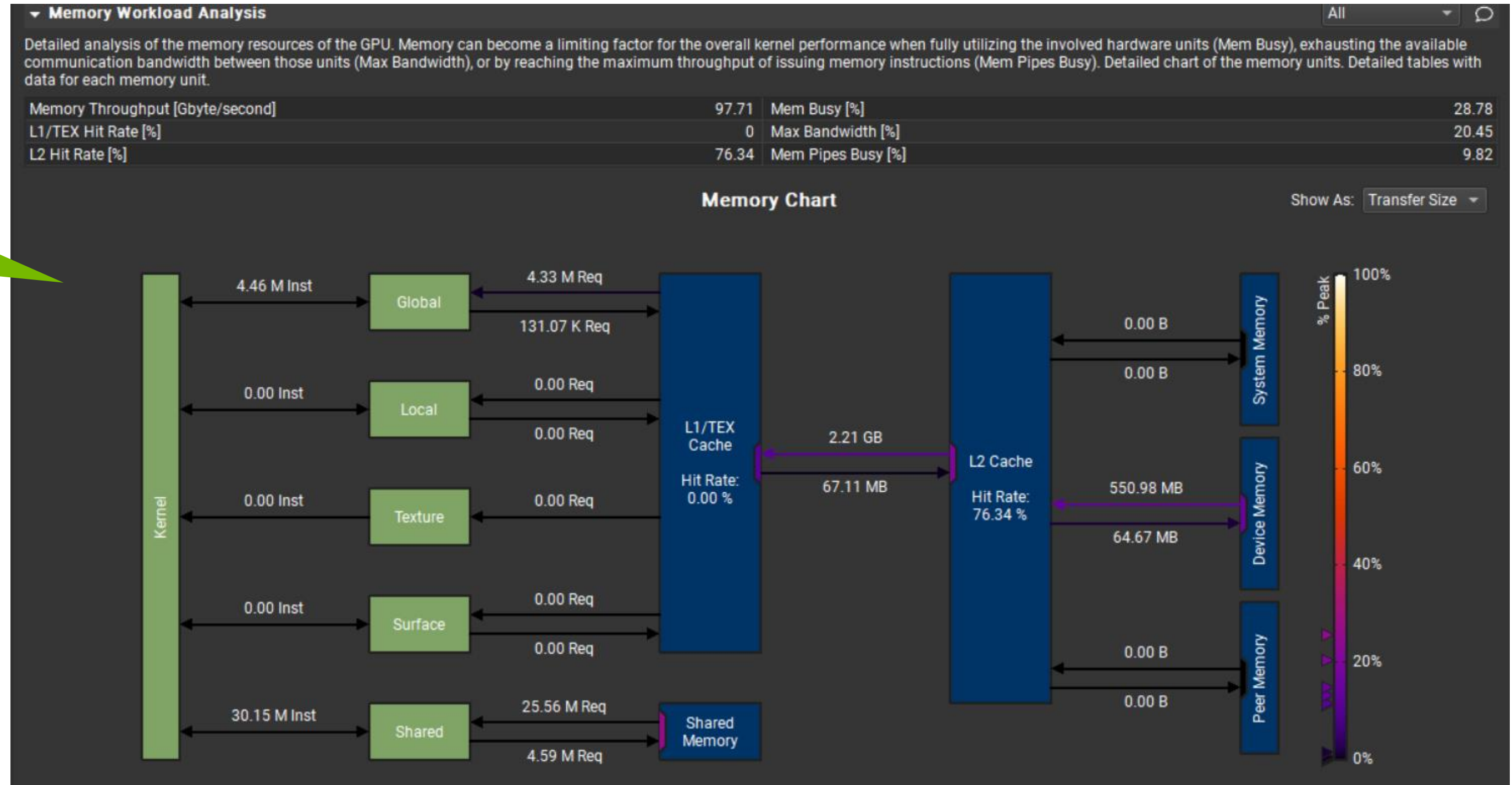
Built-in expertise for Guided Analysis and optimization

Customizable data collection and presentation



Visual memory analysis chart

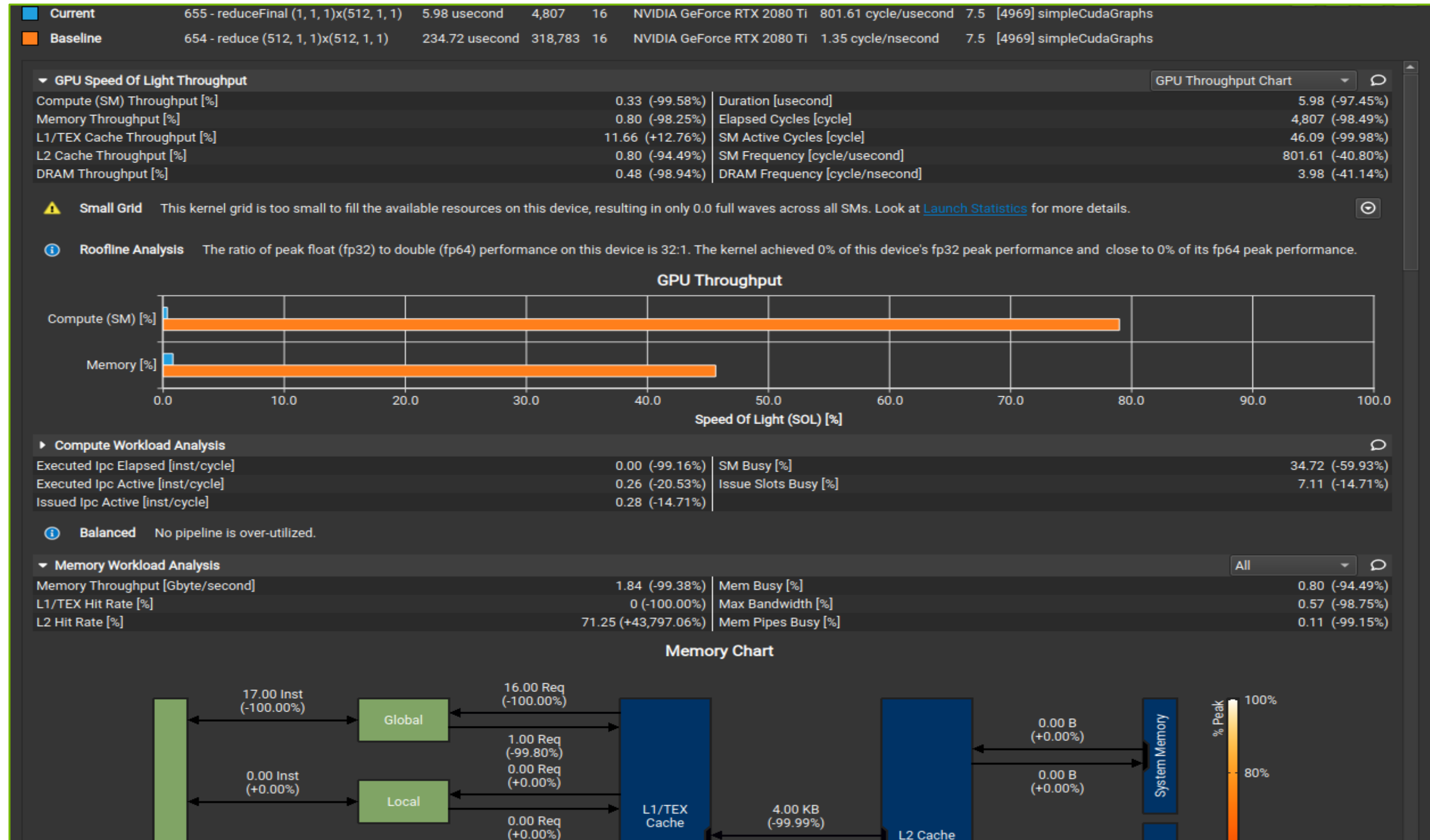
Metrics for peak performance ratios



Shared Memory					
	Instructions	Requests	Wavefronts	% Peak	Bank Conflicts
Shared Load	25,559,040	25,559,040	136,852,276	22.57	35,140,404
Shared Store	4,587,520	4,587,520	19,398,656	3.20	1,572,864
Shared Atomic	0	0	0	0	0
Other	-	-	147,957	0.02	506
Total	30,146,560	30,146,560	156,398,889	25.80	36,713,774

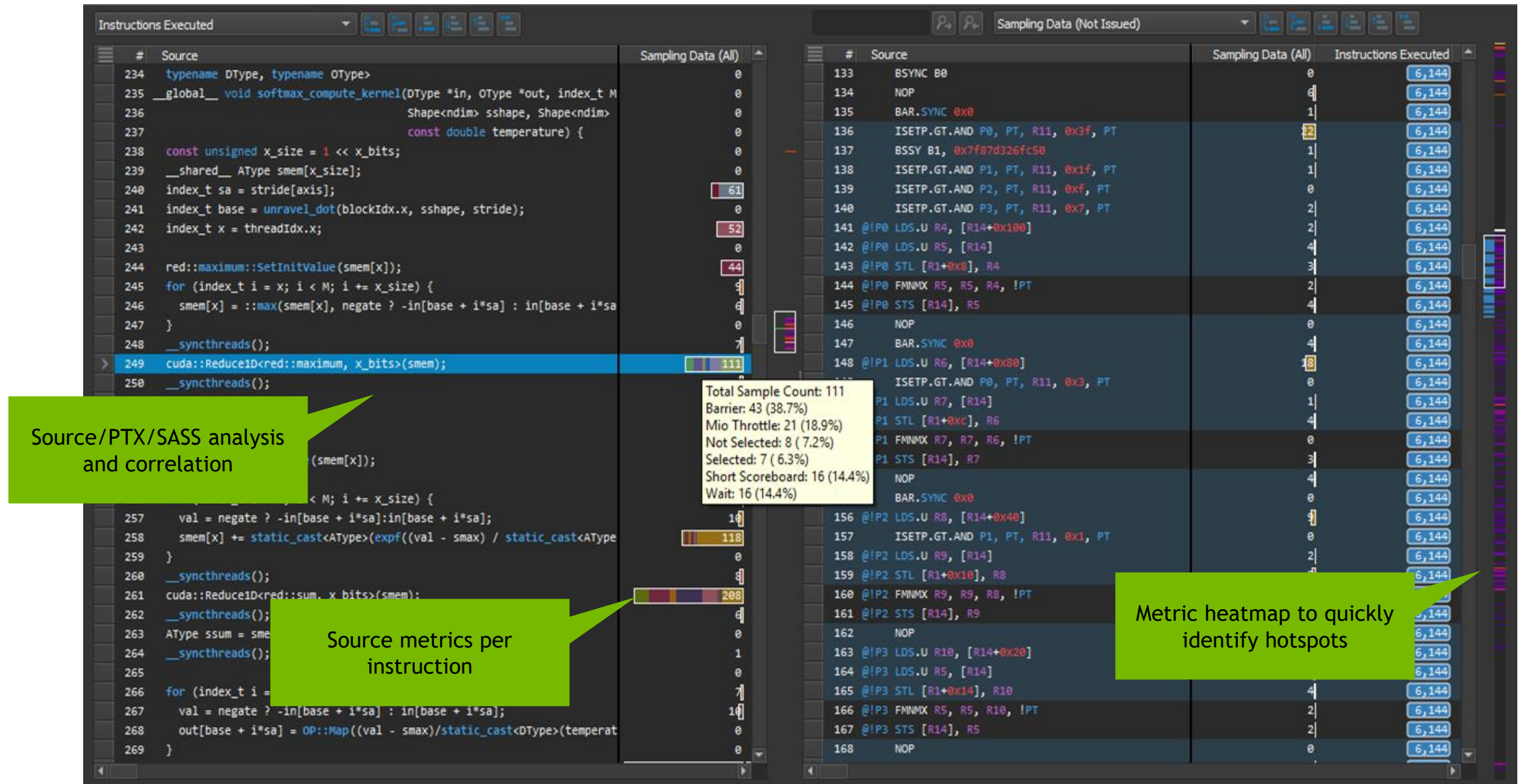
BASELINES

- Comparison of results directly within the tool with "Baselines"
- Supported across kernels, reports, and GPU architectures



SOURCE VIEW

- Source/PTX/SASS analysis and correlation
- Source metrics per instruction and aggregated (e.g. PC sampling data)
- Metric heatmap



STANDALONE SOURCE VIEWER

- View of side-by-side assembly and correlated source code for CUDA kernels
- No profile required
- Open .cubin files directly
- Helps identify compiler optimizations and inefficiencies

The screenshot displays the Standalone Source Viewer interface, which allows for a side-by-side comparison of CUDA source code and its corresponding assembly instructions. The interface is divided into two main panels.

Left Panel (Source Code):

- Page:** Source
- Launch:** 0 - 32655 - device_tea_leaf_ppcg_sol
- Time:** 1.07 msecond
- Cycles:** 1,458,003
- Regs:** 32
- GPU:** NVIDIA GeForce RTX 2080 Ti
- SM Frequency:** 1.36 cycle/nsecond
- CC:** 7.5
- Process:** [10906] tea_leaf
- View:** Source and SASS
- Source:** tea_leaf_ppcg.cuknl
- Find...** (search bar)
- Navigation:** Instructions Executed

The source code is displayed in a dark-themed editor. The current line of code is highlighted in green:

```
165 const double result = (1.0
```

Right Panel (Assembly Instructions):

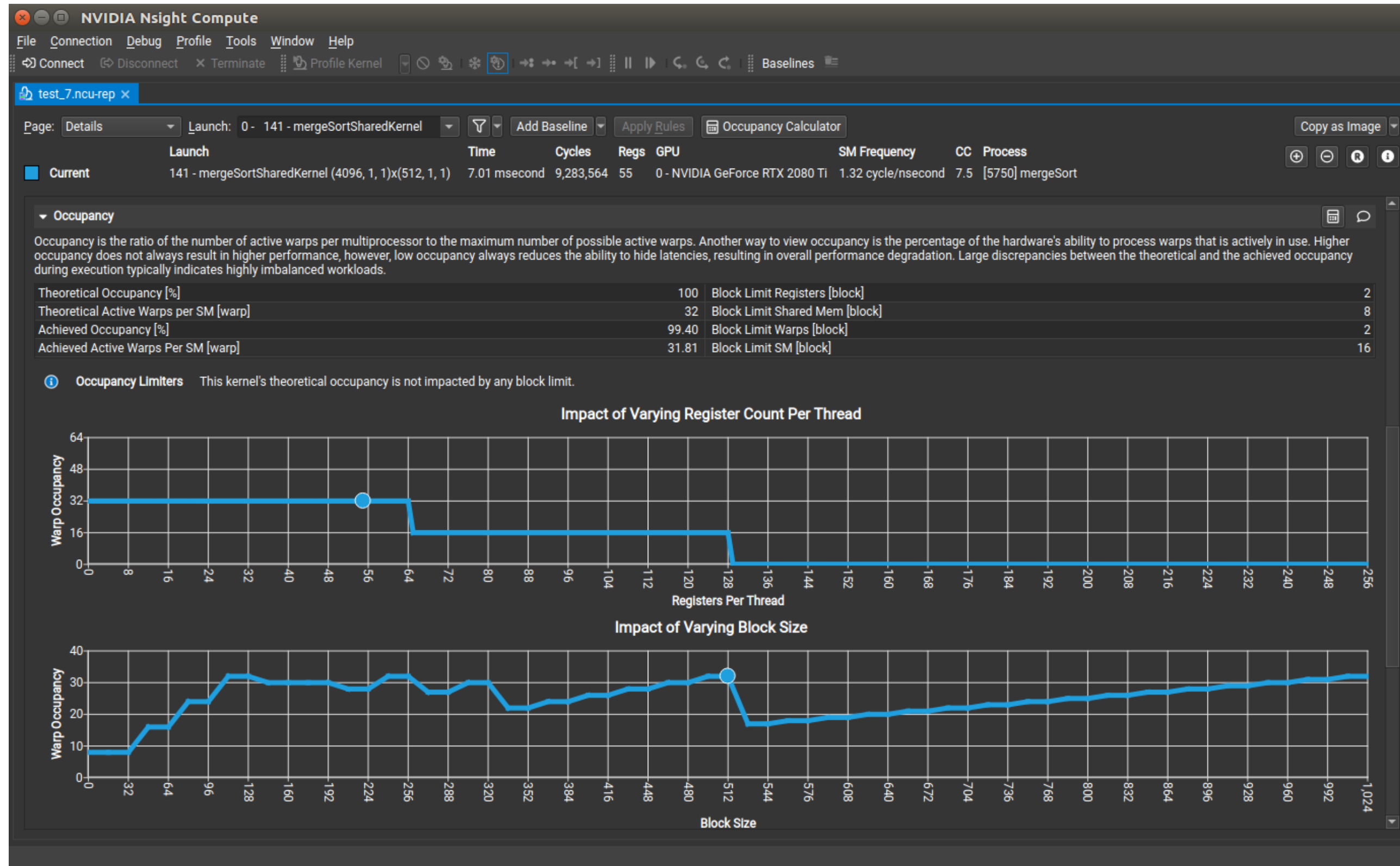
- Source:** device_tea_leaf_ppcg_solve_update_r
- Find...** (search bar)
- Navigation:** Instructions Executed

The assembly instructions are displayed in a dark-themed editor. The current instruction is highlighted in green:

```
22 00007f72 42fa6a50 IADD3 R5, R3, 0x1, RZ
```

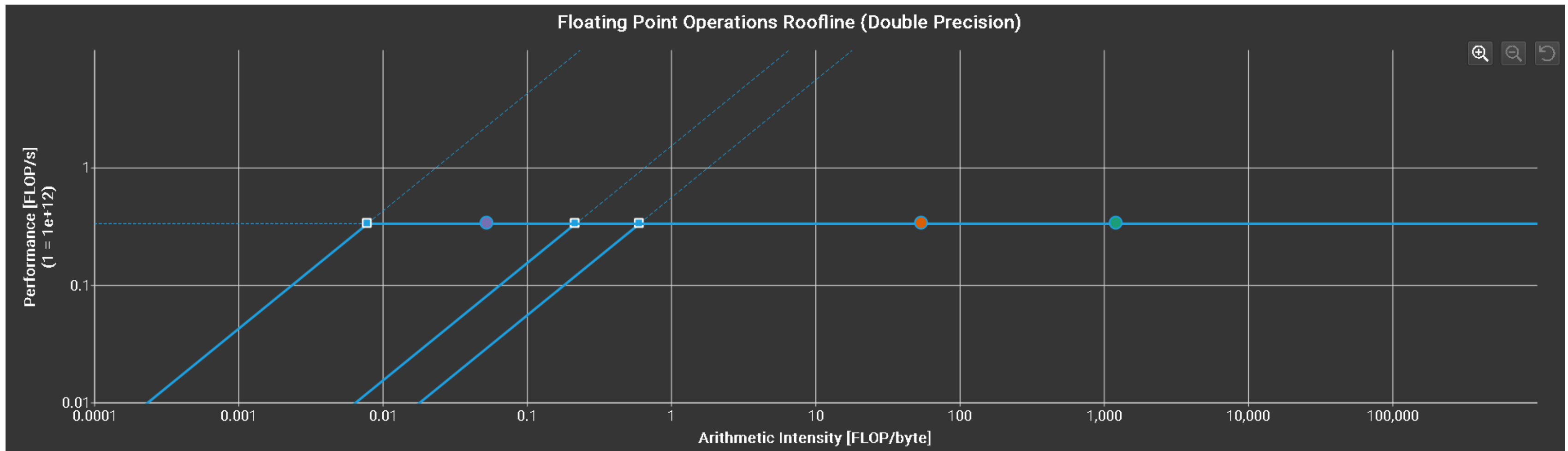

OCCUPANCY CALCULATOR

Model Hardware Usage and Identify Limiters



- Model theoretical hardware usage
- Understand limitations from hardware vs. kernel parameters
- Configure model to vary HW and kernel parameters
- Opened from an existing report or as a new activity

HIERARCHICAL ROOFLINE

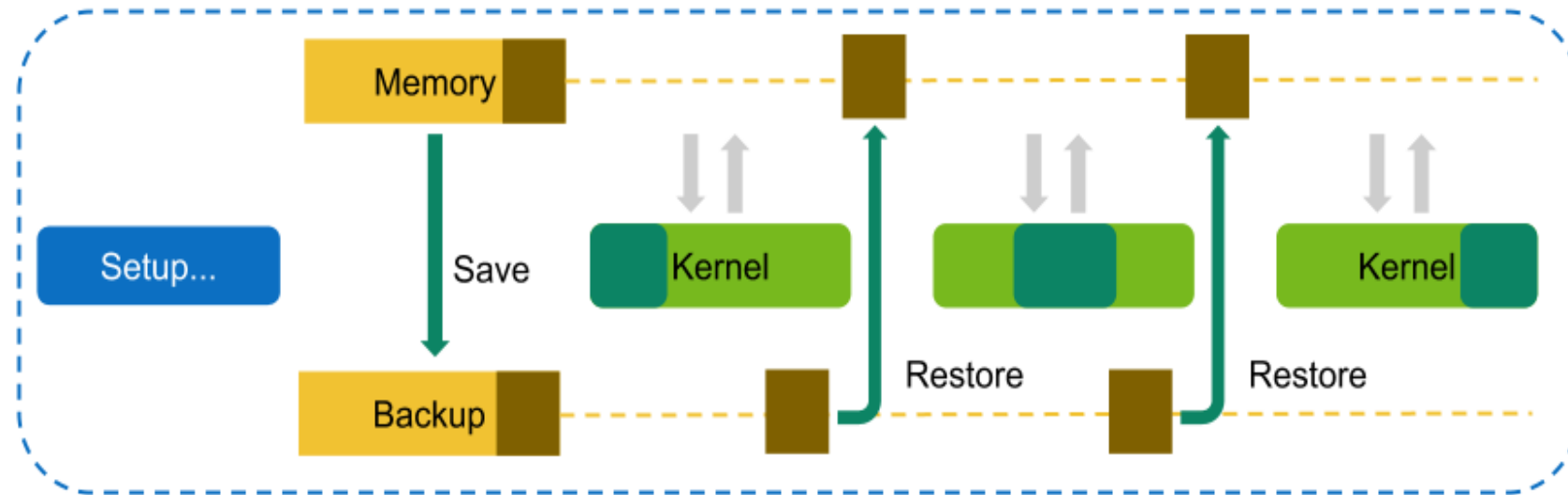


- Visualize multiple levels of the memory hierarchy
- Identify bottlenecks caused by memory limitations
- Determine how modifying algorithms may (or may not) impact performance

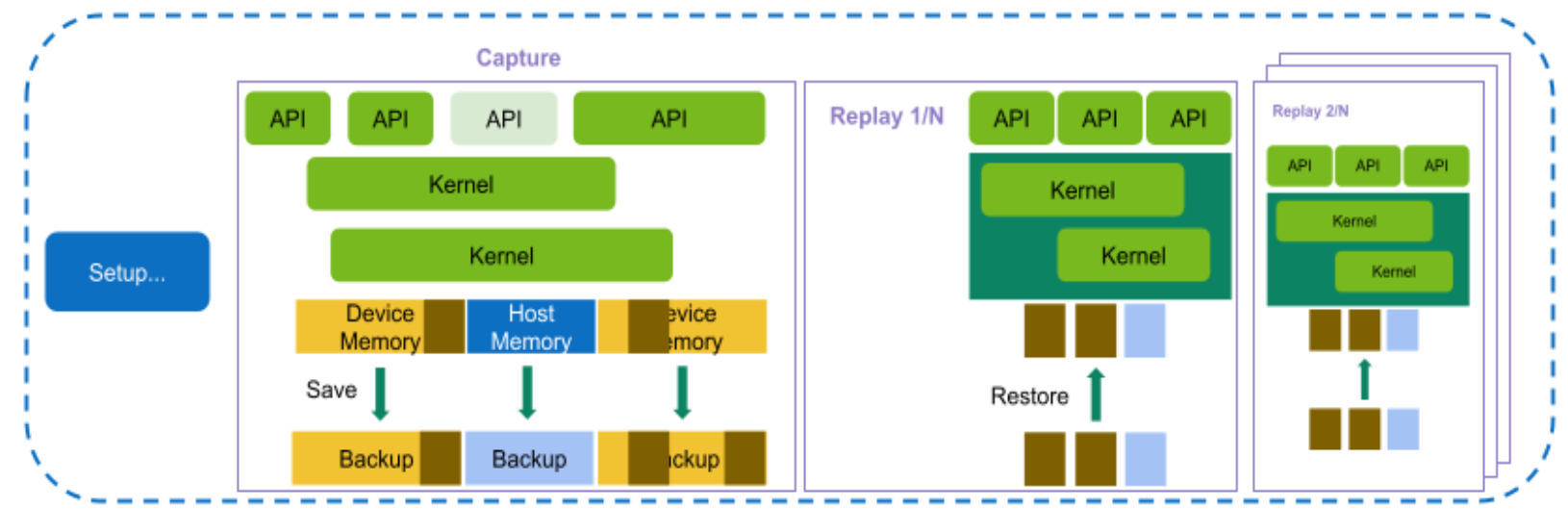
Sections/Rules Info			
Sections/Rules			
Enter filter			
	Name	Priority	Description
✓	GPU Speed Of Light Throughput (1)	10	High-level overview of the throughput for compu...
✓	GPU Speed Of Light Roofline Chart (1)	11	High-level overview of the utilization for comput...
✓	GPU Speed Of Light Hierarchical Roofline Chart (Double Precision)	12	High-level overview of the utilization for comp...
✓	GPU Speed Of Light Hierarchical Roofline Chart (Half Precision)	12	High-level overview of the utilization for comput...
✓	GPU Speed Of Light Hierarchical Roofline Chart (Single Precision)	12	High-level overview of the utilization for comput...
✓	GPU Speed Of Light Hierarchical Roofline Chart (Tensor Core)	12	High-level overview of the utilization for comput...
	Compute Workload Analysis (2)	20	Detailed analysis of the compute resources of t...

REPLAY MODES

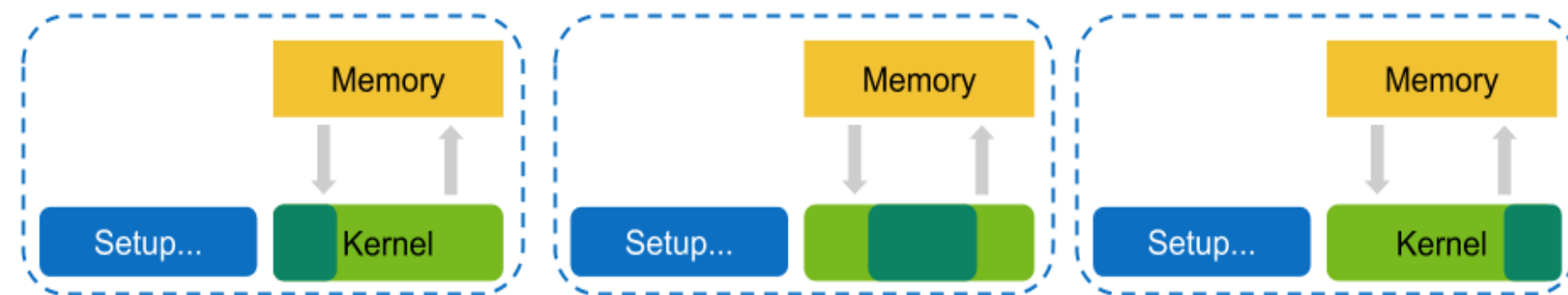
<https://docs.nvidia.com/nsight-compute/ProfilingGuide/index.html#replay>



Kernel Replay
(interactive and non-interactive)



Range Replay
(non-interactive)



Application Replay
(non-interactive)

FURTHER INFORMATION

Download

<https://developer.nvidia.com/cuda-downloads> (packaged in the CUDA Toolkit)

<https://developer.nvidia.com/nsight-systems>

<https://developer.nvidia.com/nsight-compute>

Documentation

<https://docs.nvidia.com/nsight-systems/>

<https://docs.nvidia.com/nsight-compute>

Support is available via:

<https://forums.developer.nvidia.com/c/development-tools/>

More information at:

<https://developer.nvidia.com/tools-overview>

